Constrained Aggregations

Mick Seaman

Prior presentations[1] to the P802.3ad Link Aggregation Task Force have commented both on the desirability and difficulty of rapidly and deterministically resolving which links should be included in an aggregate when one or both communicating systems have constraints on inclusion that are not easily represented by keys[2]. There is a particular and common need to be able to handle the constraint that only so many ports on a given system be aggregated[3].

The current working suggestion is that this need be met by allowing LACP implementations to dynamically modify the key values used. This is known to be problematic. The final selection of links is not necessarily deterministic, being subject to timing races in the protocol, and the time to resolve is not known. A measure of our lack of confidence in this solution is such that we feel unable to fully specify the procedure for changing keys, supposing that different sets of constraints may require different strategies. However we do recommend that keys only be dynamically changed toward more restrictive sets of ports, so the process of searching for the selected set will eventually terminate in a multi-vendor scenario, though possibly with only one link in the aggregate.

However it must be said that this procedure, unsatisfactory as it is, is probably preferable to an attempt to catalog all constraints and devise a method of communicating these in the protocol.

This note proposes an alternative way of handling constraints that does not require dynamic keys or changes to LACP[4]. It is guaranteed to resolve the inclusion of links in an aggregate within calculable bounded time. It maintains the key information for a link not included in the aggregate, allowing it to function as a "hot standby", ready to replace a failed link with the minimum of additional protocol exchanges.

[1] Luc Pariseau, Support for More Limitations, July '98. Norm Finn, Dynamic Port Keys, November '98.

[2] The single key value, assigned to a port that connects a system to a link, represents as set label assigned by that system. Ports on a given system with the same key value may be aggregated by the LACP control protocol, provided their links connect to the same partner system, and the connecting ports on that system have the same key value as each other.

[3] To be more precise, a common hardware constraint is that only a limited number of ports in the aggregate share in the transmission and reception of user data frames.

[4] It does involve being specific about the way LACP is used, and, for multi-vendor interoperability, specify a basic rule for including links in an aggregate. It does not involve adding parameters to the protocol or changing the meaning of existing parameters or existing procedures.

Proposal

LACP already contains a mechanism to cope with delays involved in organizing distributor and collector resources, or indeed the absence of the resources necessary to allocate a link to an aggregator. The Synchronization bit performs this function, essentially bridging between that part of the protocol that exchanges System ID and Key information[5], and the part that allocates and deallocates and turns on and off the transmission and reception resources.

So, if a port's key is not dynamically changed yet it is not possible to add its link to the correct aggregate, the system experiencing the difficulty would have to signal Out_of_Sync to conform to the current specification.

In addition to recognizing the opportunity that presents to resolve our difficulty, we have to agree on a deterministic algorithm for selecting amongst all those links that might be included in the aggregate. In fact this is essential once more than one vendor notices the possibilities described here[6]. I suggest that this be that ports be selected for inclusion in the aggregate based on the port numbering assigned by the system with the lower System ID, starting with the lowest numbered port as having the highest priority and working upward applying constraints. Since the

[5] And discuss how often that information should be exchanged.

[6] i.e. right now, unless we are to try and craft a complex ban.

System IDs and Keys are not changing while this process is ongoing, both systems have access to the same ordered list of ports to make their selection.

Maximum links constraints

The process described is sufficient to resolve constraints, at one or both participating systems, on the maximum number of ports that can be active in an aggregate. I believe this will probably be the most common and hence the most important constraint. This solution guarantees that the maximum number of links will actually be used in the aggregate.

Determinism, if required, can be preserved since a new link can be added to the aggregate, judged on its correct relative priority, and another link taken Out_of_sync if necessary.

Other constraints

Other constraints can be handled, as indicated above, by working up the numbered port list, applying those constraints. That is to say, if a system can make that port active in the aggregate it should do so. To ensure resolution that determination must not be based on the value of the Synchronization bit received from the system's partner.

Unlike the case of having simple constraints on the maximum number of links to be included in an aggregate, this is not a completely satisfactory solution. That is to say links may be held idle when knowledge of the total situation might have arranged them better. So we may want to hold onto the idea of dynamic key changing[7]. If we do I would suggest the following recommendation as a somewhat less "all bets are off" enhancement to the current recommendation[8].

The system with the higher numbered System ID ought not to attempt to dynamically change its key values. The system with the lower numbered System ID, effectively the Master in this relationship, may choose to change the keys for each port. Key changes should be consistent with maintaining its lowest numbered port in the aggregate as an active[9] link.

Successive key changes, if they occur, should progressively reduce the number of ports in the original aggregate[10]. The original key should be maintained for the lowest numbered thought to be aggregatable. Key changes should not be made more often than once in any 60 second interval.

[7] At the very least we should make a definite statement on the subject, banning it if we mean to ban it, adopting Norm's rule or some further refinement as proposed here otherwise.

[8] If we want to allow key changes at all I suggest that the mandatory constraint be no more specific than it is at present.

[9] In_sync

[10] Not absolutely required now changes are being made only at one end, a better search algorithm might be found. A binary chopped slave?

These recommendations are basically equivalent to giving the master permission to search the space of possibilities trying to find the best proposition to make to the slave, watching the slave's slow reaction by looking at the slave's willingness to synchronize on each proposition. Not a pretty sight perhaps, and I wouldn't want to build it, but better than leaving it all up in the air. At least I could successively interoperate with such an implementation.

Standby links

Ports that are not active are held Out_of_Sync. If it is desirable to hold them as hot standbys this is a win. If there is more than one such port, and their keys would have otherwise been changed to allow them to participate in another aggregate, the desirability of the outcome depends on how that aggregate would have been used by higher layer protocols. If it is fact was to be a blocked link in the spanning tree we are no worse off, if it would have actually been used by routing to load share we may be.

Having a rule that allows links to be held as hot standby links in this way opens up the possibility of using LACP even when the physical hardware on one or other system does not support distribution or collection to more than one link. It allows a parallel links to be autoconfigured as a hot standby[11] and deployed to mask a failure without disrupting higher layer protocols at all.

[11] Which the naïve user might well have expected all along.